4. Consider the array x3d defined as follows:

```
int x3d[3][5][7];
```

Consider the element x3d[1][1][1]. The element of the array in the next (sequential) int-sized storage location is

(a) `x3d[2][1][1]`

(b) `x3d[1][2][1]`

(c) `x3d[1][1][2]`

(d) `(x3d[2][1][1])+1`

(e) None of the above

5. If a programmer wishes to use the standard library function *strcat()* in her or his program, which of the following statements should appear near the beginning of the program?

(a) `#include <stdio.h>`

(b) `#include <string.h>`

(c) `#include <stdlib.h>`

(d) `#include "stdio.h"`

(e) All of the above

6. Which of the variable types below can hold the largest range of values?

(a) `char`

(b) `int`

(c) `long int`

(d) `float`

(e) None of the above

7. Which of the following is a storage class specifier?

(a) `auto`

(b) `register`

(c) `static`

(d) `extern`

(e) All of the above

8. Which of the following is an interactive debugger on UNIX for C programs?

(a) *cpp*

(b) *make*

(c) *gdb*

(d) *lint*

(e) All of the above

A. ***(8 marks)***

For each of the following multiple-choice (mystical-guess?) questions, give the best answer.

1. Which of the following data types cannot be the type of the return value of a function?

(a) `int`

(b) `char`

(c) `float`

(d) double

(e) `void`

(f) pointer (to any valid type)

(g) None of the above

2. Consider the following declaration

```
struct complex {
    double re;
    double im;
};
```

Assume that within main() we have the follow statements:

```
main()
{
   struct complex *a, b;

   a = &b;
   a->re = 3.0;
   a->im = 7.4;
   print_complex( a );
}
```

Which of the following expressions is equivalent to a->re ?

(a) `*a.re`

(b) `(*a).re`

(c) `*(a.re)`

(d) `b->re`

(e) None of the above

3. What is the role of a typedef statement?

(a) Allows a programmer to explicitly associate a type with an identifier.

(b) Allows a programmer to define new, structured types.

(c) Allows a programmer to override the meaning of a built-in type.

(d) Allows a programmer to set the type default for variables which are not explicitly declared.

(e) None of the above

B. (5 marks)

There are many useful options to the command which invokes the *cc* or *gcc* compiler. A number of those options are listed below. For each, say what it is used for; i.e. what it controls.

(a) -O

(b) -Dname=def

(c) -c

(d) -o

(e) -Wall

C. (1+1+1+1 = 4marks)

Given the following variable definitions

```
int i1, i2;
float f1;
```

give a C statement that will achieve each of the following. You can assume an architecture in which `int` and `float` values are 32 bits in size.

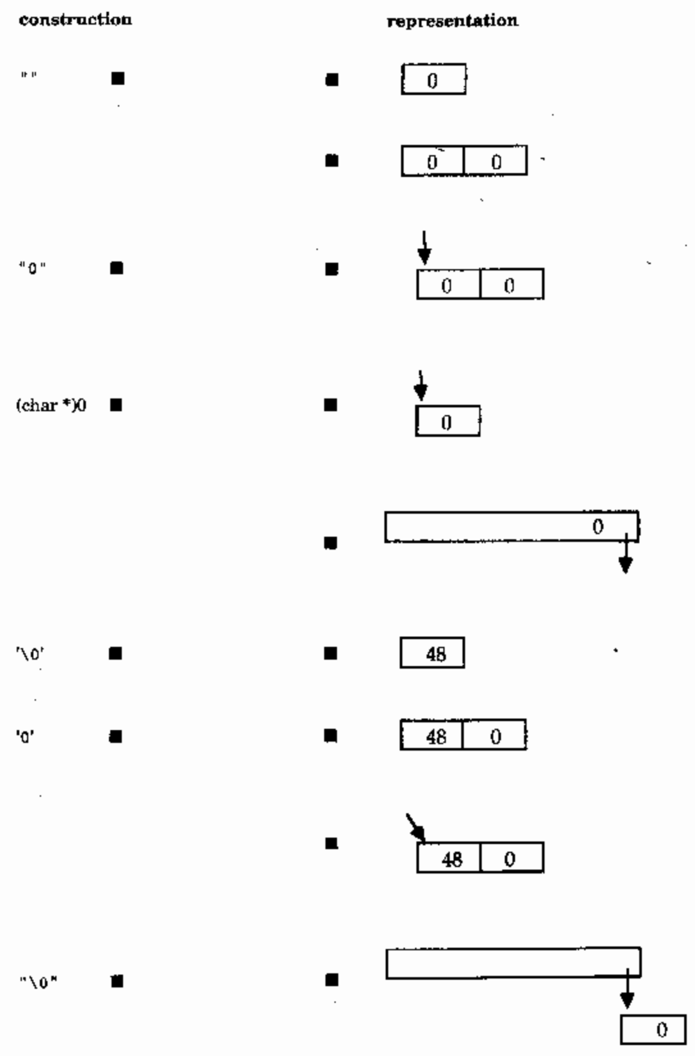
1. Calculate the remainder when `i1` is divided by `i2`.
2. Finding the remainder when `i1` is divided by the value of `f1`, treating `f1` as an integer.
3. Turning off (setting to 0) the lower four bits in `i1`.
4. Turning on (setting to 1) the upper four bits in `i2`.

D. (6 marks)

Consider each of the following constructions involving data of type `char`. Match each construction on the left with the correct diagram on the right. Use a straight line to indicate the match. That is, draw a straight line between each expression or string and its corresponding pictoral explanation. In the diagrams arrows indicate pointers, and the size of rectangles ("boxes") is directly related to the size of the memory location storing the value indicated within the rectangle. Note that there are more possible representations than constructions; i.e. not every representation will be matched with a string or expression.

You can assume that the word size is 32 bits. Recall that the ASCII code for the '0' character is 060 (octal) or

48 (decimal).



5. Suppose a C program contains a declaration

```
int x, y;
```

and later the statement

```
y = ( x == 3 ? 7 : y-1 );
```

Rewrite this last statement as an if-then-else statement. Make sure that the statement you give has the same effect as the statement above.

6. Given the following function

```
void funky( int *ptr ) {
    static int dummy = 5;
    ptr = &dummy;
}
```

which is called from main() as follows:

```
int main() {
    int *iptr, x;

    x = 10;
    iptr = &x;
    funky( iptr );
}
```

What does iptr point to after returning from the function call and what is the final value of x?

7. Give an example of a conditional macro definition.

Note: the definition must be completely correct for full marks.

E. (3+1+2+1+2+2+2 = 13 marks)

For each of the following questions give a short, precise answer.

1. Given

```
char s[] = "C World";
const char *cptr = s;
```

state whether or not each of the following statements is legal. (Your answer should be either "legal" or "illegal").

i) `cptr++;`

ii) `cptr[0] = 'B';`

iii) `s[0] = 'B';`

2. Consider the following program fragment:

```
int a, b, c;
a=100; b=20; c=3;
printf( "%d", a/b/c );
```

What gets printed on the standard output?

3. Consider the following infinite loop in C:

```
while( 1 )
{
...
  statement
...
}
```

Give two different statements which could be used in place of *statement* to exit such a loop.

4. What value or values are treated as "true" in C?

G. ***(2+4 = 6 marks)***

Answer each of the following questions with a concise answer.

1. What is the difference between a *declaration* and a *definition* in the context of variables in C? Give an example which highlights the difference between the two.

2. What is the main difference between a union and a `struct`? How are they similar? Use examples to illustrate.

F. ***(5+3 = 8 marks)***

1. One of the program examples given in the C Short Course was a program which performed functionality similar to the `cat(1)` command. The code for the program is given below. However, various names, symbols, operators, and operands have been replaced by instances of ______ (a blank). Fill in each of the blanks with the correct name, symbol, operator, or operand. Note that each blank stands for one, and only one, name, operator, operand, or symbol.

```
#include <stdio.h>
int main( ______ argc, ______ *argv[] )
{
    ______ *fp;
    char char_buf[512];
    int counter;

    if( argc != 2 )
    {
        fprintf( stderr, "Usage: %s filename.\n", argv[0] );
        exit( 1 );
    }
    else
    {
        if( (fp = fopen( argv[1], "r" )) == NULL )
        {
            fprintf( stderr, "Cannot open %s file.\n", argv[1] );
            exit( 2 );
        }
    }

    counter = 0;
    while( fgets( char_buf, 512, fp ) )
    {
        ______( stdout, char_buf );    /* output the string */
        counter______;     /* increment counter by 1 */
    }

    fprintf( stdout, "\n Total lines: %d\n", counter );
    fclose( fp );
    return( 0 );
}
```

2. Indicate how you would modify the code above to make use of dynamically-allocated storage for the buffer to hold the characters read from the input file. Make sure you show what you would remove, add, remove, or change.

   You can indicate your answer in the space below, or as modifications marked on the code above.

Supplementary information

You may find the following function prototypes useful in answering some of the questions in this exam:

```
int fclose( FILE *stream );
int fgetc( FILE *stream );
char *fgets( char *s, int size, FILE *stream );
FILE *fopen( const char *path, const char *mode );
int fprintf( FILE *stream, const char *format, ... );
int fputc( int c, FILE *stream );
int fputs( const char *str, FILE *stream );
void free(void *ptr);
int getc( FILE *stream );
int getchar();
char *gets( char *str );
void *malloc(size_t size);
int open( const char *path, int flags, mode_t mode);
void perror( const char *string );
int printf( const char *format, ... );
int putc( int c, FILE *stream );
int putchar( int c );
int puts( const char *str );
void *realloc(void *ptr, size_t size);
int scanf( const char *format, ... );
int sprintf( char *str, const char *format, ...);
```

Extra Space

(The space below is for answering previous questions or for rough work.)